

DOSSIER DE PROJET

Rabillard Fabien

Titre professionnel DWWM

Développeur Web & Web Mobile

Application Web Hypnos

Sommaire

Introduction

Liste des activités types et des compétences professionnelles	page 3
Résumé du dossier de projet	page 4

Cahier des charges,besoins ou spécifications fonctionnelles

Objectifs du projet	page 5
Cahier des charges et besoins du client	page 5
Spécifications technique du projet	page 7
Environnement de développement	page 8
Déploiement	page 8
Symfony	page 8
Sécurité	page 9

Réalisations

Diagramme de cas d'utilisation	page 10
Wireframes	page 10
Charte graphique	page 12
Conception de la base de données	page 13
Initialisation du projet	page 14
Création de la base de donnée	page 15
Création des entités	page 15
Authentification des utilisateurs	page 16
Sécurité de la base de donnée	page 17
Back-Office	page 18
Front-end	page 20
Script Javascript	page 21
Manuel d'utilisation	page 23
Référencement	page 24

Jeu d'essai

page 24

Veille de sécurité

page 27

Recherche et traduction anglophone

page 28

Conclusion

page 29

Annexes

page 31

Introduction

1 - Liste des activités types et des compétences professionnelles

1. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

2. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

Résumé du dossier de projet

Durant ma formation de développeur web et web mobile (DWWM) chez Studi, une évaluation en cours de formation a eu lieu du 14/03 au 21/04/2022.

C'est dans ce cadre que ce projet fictif a été réalisé.

Hypnos est un groupe hôtelier composé de 7 établissements à travers la France.

Afin de ne plus dépendre de sites tiers comme booking.com ou trivago.fr pour la location de ses chambres, le groupe hôtelier souhaite disposer de son propre système de réservation et d'un site web dédié.

Les principales fonctionnalités qui ont été demandées par Hypnos :

Chaque établissement à sa propre page web où sont présentées ses suites avec ses informations.

Les visiteurs peuvent voir l'intégralité du catalogue et contacter le groupe hôtelier via un formulaire dédié.

Néanmoins, ils devront s'enregistrer ou se connecter pour pouvoir accéder au formulaire de réservation.

Ils auront alors aussi accès à leur historique et pourront annuler une réservation future si besoin. Par contre, l'annulation doit avoir lieu au maximum 3 jours avant le début de la réservation.

Le maintien de l'application se fera par un administrateur (employé d'Hypnos) qui aura aussi la charge de créer, modifier ou supprimer les établissements ainsi que de maintenir et d'affecter un gérant à un établissement.

Le gérant ne s'occupe que de l'établissement où il a été affecté, et se charge d'indiquer et gérer les différentes suites de son hôtel.

Après étude du dossier et des fonctionnalités demandées, j'ai choisi d'utiliser PHP et Symfony pour la partie back-end et HTML/CSS et Bootstrap via Twig pour la partie front-end.

Symfony ayant l'avantage d'être fourni avec énormément d'outils de sécurité, et de bundles permettant de créer une application simple et sécurisée qui correspond à la demande du client.

Le «security bundle» propose des outils comme l'encodage de mot de passe, une authentification renforcée, et la gestion des rôles qui permettent de définir le niveau d'accès des utilisateurs connectés.

Symfony dispose aussi d'un moteur de template (Twig) qui facilite l'intégration front-end ainsi que la communication avec le back-end.

Cahier des charges,besoins ou spécifications fonctionnelles

Objectifs du projet

L'objectif du projet est de concevoir et créer une application web responsive suivant les exigences et besoins du client exprimés dans le cahier des charges.

Pour cela il convient de réfléchir et élaborer :

- Une charte graphique cohérente
- Des diagrammes UML (cas d'utilisation, de séquence,...)
- Un manuel d'utilisation de l'application

Cahier des charges et besoins du client

Hypnos souhaite disposer de son propre système de réservation avec un nouveau site web dédié, Plusieurs fonctionnalités étaient demandées par le client :

1- Gestion des établissements :

Un administrateur (employé d'Hypnos) sera chargé du maintien de l'application web.

Il est le seul autorisé à créer, modifier et supprimer des établissements.

Chaque établissement dispose d'un nom, d'une adresse , d'une ville et d'une description.

Il est aussi le seul à pouvoir assigner et gérer les responsables des établissements : les gérants.

Comme l'administrateur, le gérant à un nom, un prénom, une adresse email et un mot de passe sécurisé.

2- Gestion des suites

Le gérant ne s'occupe que de l'établissement où il a été affecté.

Depuis son interface utilisateur, il peut indiquer une ou plusieurs suites et les gérer (modification ou suppression).

Chaque suite possède un titre, une image en avant, une description, son prix (fixe tout au long de l'année) une galerie d'images, un lien vers sa réservation booking.com.

3- Site web

Les visiteurs ont accès à l'ensemble des établissements et des suites du groupe Hypnos.

Chaque établissement dispose de sa propre page où sont indiquées, en plus des infos de l'hôtel, les différentes suites et leurs informations.

4- Formulaire de contact

Un visiteur ou un client doit pouvoir contacter le groupe Hypnos pour avoir plus d'informations ou commander un service supplémentaire à la prestation initiale.

Plusieurs informations sont demandées comme le nom/prénom et email de la personne ainsi qu'un sujet prédéfini et un corps de message.

Les 4 sujets sont :

- Je souhaite déposer une réclamation
- Je souhaite commander un service supplémentaire
- Je souhaite en savoir plus sur une suite
- J'ai un souci avec cette applications

Dans un premier temps, ce sera l'administrateur qui sera unique destinataire des messages.

5-Réservations des suites

Il sera possible de réserver une suite via une page dédiée du site web.

Néanmoins, cette possibilité est réservée aux seuls utilisateurs inscrits (clients).

Pour cela il faut se connecter ou s'inscrire sur le site.

Pour l'inscription seront demandés :

un nom, un prénom, une adresse email et un mot de passe sécurisé.

Sur le formulaire de réservation, 4 champs de formulaires seront présents :

Le choix de l'hôtel, le choix de la suite, la date de début du séjour et la date de fin.

Le client doit pouvoir, sans rechargement de la page, savoir si la suite est disponible suivant les dates indiquées.

Outre le fait de pouvoir réserver une suite, le client connecté pourra accéder à une page contenant l'historique de toutes ses réservations.

Il pourra aussi annuler une réservation, à condition de la faire maximum 3 jours avant la date de début du séjour.

6-Expérience utilisateur

Afin de gagner en expérience utilisateur, et faire gagner du temps au client,lorsqu'il clique sur le bouton «réserver» situer sur la page listant les suites, il sera redirigé vers le formulaire de réservations et les champs «hôtel» et «suite» du formulaire devront être pré-remplis.

Spécifications techniques du projet

Le projet ne demandant pas de fonctionnalités front-end poussées, j'ai opté pour Symfony en back-end couplé avec un combo HTML5 / Bootstrap et Twig.

Bootstrap permet de disposer d'outils, de composants prêt à l'emploi et d'une feuille de style pré-existante. Sur ce projet, c'est la version scss d'un thème différent du thème par défaut de Bootstrap qui a été utilisée, afin de pouvoir personnaliser et intégrer la charte graphique. (Thème Simplex par Bootswatch)

Pour le formulaire de réservation, j'ai utilisé JavaScript et Jquery pour éviter le rechargeement de la page afin de ne proposer que les suites correspondantes à l'établissement choisi par le client.

Même si ce projet est fictif, j'ai utilisé l'application Trello pour planifier mes tâches.

Cela m'a permis d'avoir une vue d'ensemble de l'entièreté du projet

J'ai séparé mes tâches avec plusieurs colonnes :

- "Coding Zone" et "No Coding Zone" pour le backlog.
- "Livrables" qui est la liste de tous les documents à rendre avec le projet.

Et les 2 colonnes usuelles "en cours" et "terminé".

Pour le développement, afin d'avoir un suivi de mon code, j'ai utilisé Git (v 2.34.1) comme logiciel de versionning.

Les avantages de Git sont multiples comme par exemple :

- La possibilité de revenir en arrière si jamais les dernières modifications ne sont pas correctes
- La possibilité de développer et tester de nouvelles fonctionnalités sans impacter l'environnement de production via la création de branches.

J'ai travaillé avec 2 types de branches en particulier:

- La branche "main" qui contenait la dernière version fonctionnelle du code
- Diverses branches qui souvent correspondent à la fonctionnalité en cours de développement

Une fois la fonctionnalité codée et testée, je la fusionne sur la branche "main".

Afin d'avoir une sauvegarde de mon projet en ligne, une visualisation plus facile de mon code et des modifications, j'ai utilisé GitHub.

Environnement de développement:

Lors du développement, j'ai utilisé Ubuntu 21.10 qui est mon système d'exploitation principal. J'ai aussi utilisé une machine virtuelle Windows 10 via le logiciel VirtualBox afin de pouvoir utiliser la suite Adobe.

L'IDE utilisé sur ce projet est PHPStorm.

J'ai choisi cet IDE, dont une licence nous est offerte avec la formation, car cet éditeur de code est vraiment pensé pour PHP et Symfony.

Il offre des outils de saisie semi-automatique et de refactorisation de code, de débogage mais aussi l'accès au terminal ou à des outils de gestion de la base de données, le tout directement intégré au logiciel.

En local, j'ai utilisé :

- Mysql (version 8.0.28)
- PHP 8 (version 8.0.17)
- Apache (version 2.4.48)
- Node.JS (version 6.13.2)

Déploiement:

Pour le déploiement, j'ai choisi Heroku, une plateforme d'hébergement reconnue qui dispose de normes de sécurité de haut niveau.

De base, il implémente de base le protocole HTTPS qui protège l'intégrité et la confidentialité des données lors du transfert d'informations entre l'ordinateur de l'utilisateur et le site.

Grâce à Heroku CLI, j'ai pu déployer et mettre à jour mon application en ligne directement depuis le terminal de mon application.

Symfony:

Symfony ayant l'avantage d'être fourni avec énormément d'outils de sécurité, et de bundles permettant de créer une application simple et sécurisée qui correspond à la demande du client.

Il dispose aussi d'un moteur de template (Twig) qui facilite l'intégration Front-end ainsi que la communication avec le back-end.

On peut facilement analyser et améliorer l'application grâce au "Profiler" de Symfony.

Il s'agit d'une barre située en bas de page pendant la phase de développement et permet d'avoir des informations sur les éventuels erreurs, voir les logs, connaître l'utilisateur connecté ou intercepter les mails ou requêtes.

J'ai utilisé Composer comme gestionnaire de dépendances. Il permet d'installer et de mettre à jour des

dépendances dans le projet en cours.

Pour installer et gérer les dépendances JavaScript et CSS, j'ai utilisé NPM et Webpack Encore.

J'ai utilisé la version 6.06 de Symfony ainsi que certains bundles :

- Easyadmin
- Doctrine
- Security
- Twig
- Validator
- VichBundle
- ...

Sécurité:

Concernant la sécurité :

- Le back-Office administrateur et l'accès aux comptes de l'ensemble des utilisateurs sont protégés par l'authentification (via le bundle security de Symfony)
- L'accès aux différentes pages est protégés par un système d'autorisation via les ROLE
- Les mots de passes sont hasher en base de données, aucun mot de passe en clair doit se trouver en BDD
- Twig (le moteur de template) protège des injections XSS en échappant automatiquement le texte pouvant être tapé dans un formulaire
- Le serveur de production doit implémenter les standards de sécurité comme HTTPS.

Réalisations

Diagramme de cas d'utilisation:

Après avoir étudié le cahier des charges et les fonctionnalités demandées par le client, j'ai réalisé un diagramme de cas d'utilisation.

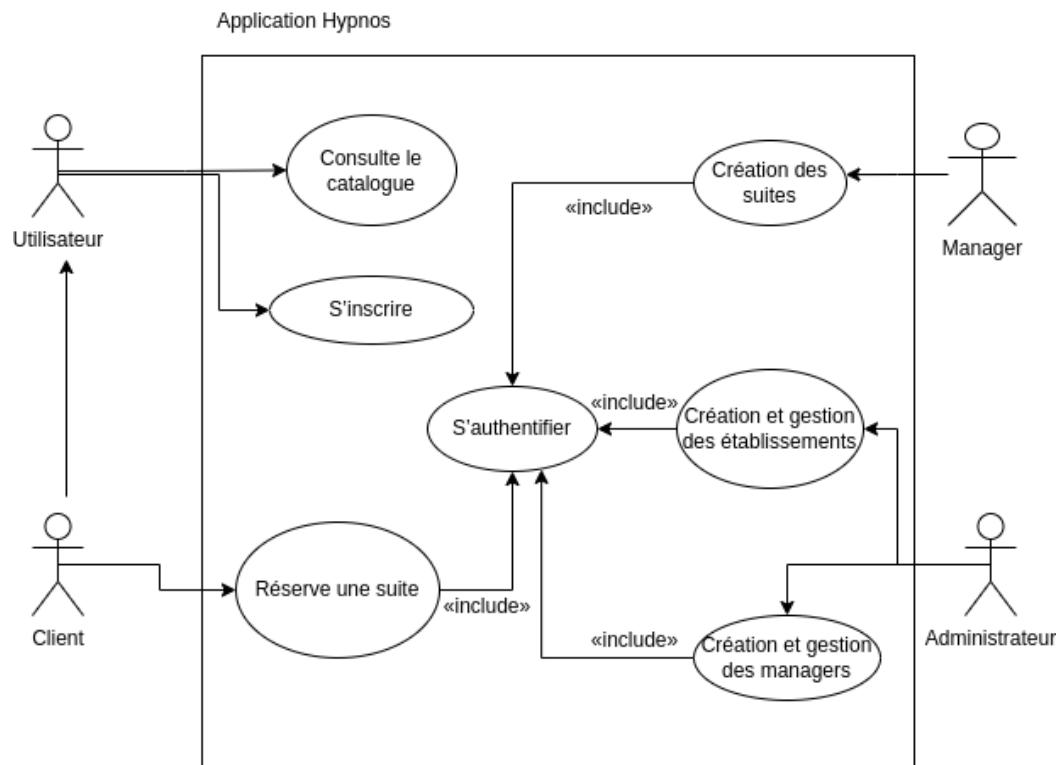


Diagramme de cas d'utilisation

Wireframes :

Ce diagramme m'a aidé ensuite de faire des wireframes des principales pages de fonctionnalités de l'application.

Hypnos

Accueil Nos établissements Mon compte Réservation Contact Déconnexion

Formulaire de réservation

Choisissez un hôtel

Choisissez une suite

Date de début dd/mm/yyyy

Date de fin dd/mm/yyyy

Hypnos © 2022
Lorem ipsum sit delém.
RGPD | CGV

Hypnos

Formulaire de réservation

Choisissez un hôtel

Choisissez une suite

Date de début dd/mm/yyyy

Date de fin dd/mm/yyyy

Hypnos © 2022
Lorem ipsum sit delém.
RGPD | CGV

Hypnos

Accueil Nos établissements Réservation Contact

Formulaire de contact

Sed sunt quia sed quae ipsum quia ut nulla nostrum. Sed enim querat eos accusamus dignissimos et galisum accusamus et doloremque laborum.

Votre nom

Votre prénom

Votre email

Sujet du message

Votre message

Hypnos © 2022
Lorem ipsum sit delém.
RGPD | CGV

Hypnos

Formulaire de contact

Sed sunt quia sed quae ipsum quia ut nulla nostrum. Sed enim querat eos accusamus dignissimos et galisum accusamus et doloremque laborum.

Votre nom

Votre prénom

Votre email

Sujet du message

Votre message

Hypnos © 2022
Lorem ipsum sit delém.
RGPD | CGV

Charte graphique

En parallèles de la création des wireframes, j'ai recherché des polices d'écriture et une palette de couleurs cohérentes que j'ai ensuite intégrés sur un mockup afin de peaufiner et valider l'ensemble et d'avoir une idée du rendu final

Police d'écriture :

Philosopher Regular

Philosopher Bold

Philosopher Italic

Philosopher Bold Italic

Proza Libre Regular

Proza Libre Bold

Proza Libre Italic

Proza Libre Bold Italic

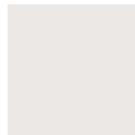
Quo error inventore quia est quos vitae.

Font Philosopher pour les titres

 Lorem ipsum dolor sit amet. Rem sint dolor ad quaerat maiores a iusto deserunt. Hic delectus dolorem et delectus tempore et molestiae cupiditate. Et blanditiis sint est fuga harum aut nisi minima ut enim totam a voluptate fugit. Et similique accusantium qui culpa repellendus et dolorem deserunt sed cumque ipsum aut distinctio enim qui adipisci odit. Aut officiis veniam sed consequatur perferendis et iusto itaque. Aut laborum illum eos excepturi iure eos temporibus similique et adipisci quia est quaerat nesciunt est repudiandae ducimus. Aut temporibus consequuntur qui officia nostrum et harum error quo dignissimos veritatis et itaque nemo et facilis dolores a incidunt necessitatibus.

Font Proza Libre pour les textes

Palette de couleurs :



#ECE8E5



#254E70



#0FA3B1



#912F56



Conception de la base de données

A l'aide du diagramme de cas d'utilisation et du cahier des charges, je me suis posé la question de savoir quelles entités j'allais avoir dans mon application.

Par exemple, un client dispose d'un email, d'un mot de passe sécurisé, d'un nom et d'un prénom et d'un rôle qui sera utilisé plus tard pour la gestion des accès dans Symfony.

De cette réflexion en a découlé un diagramme de classe :

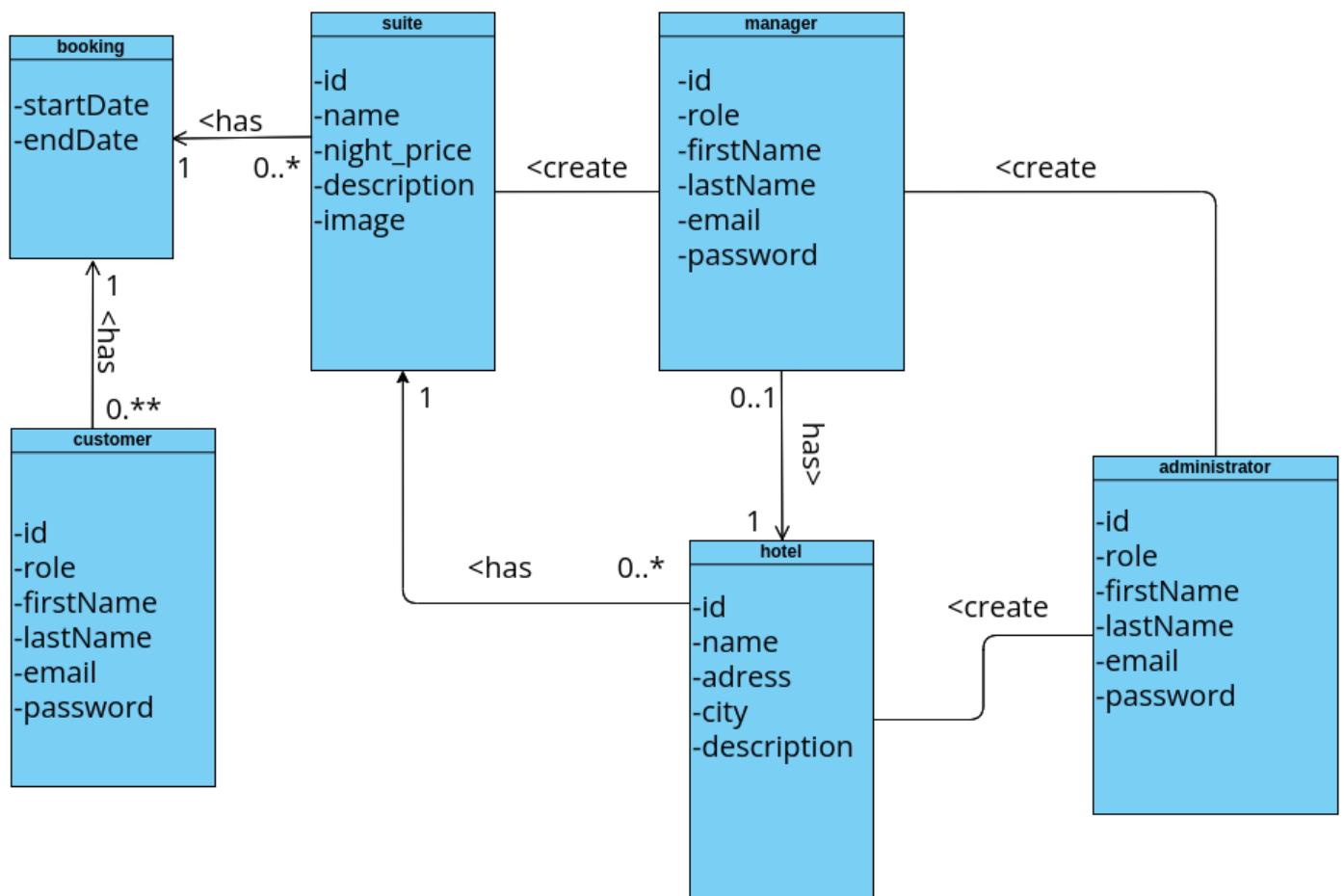
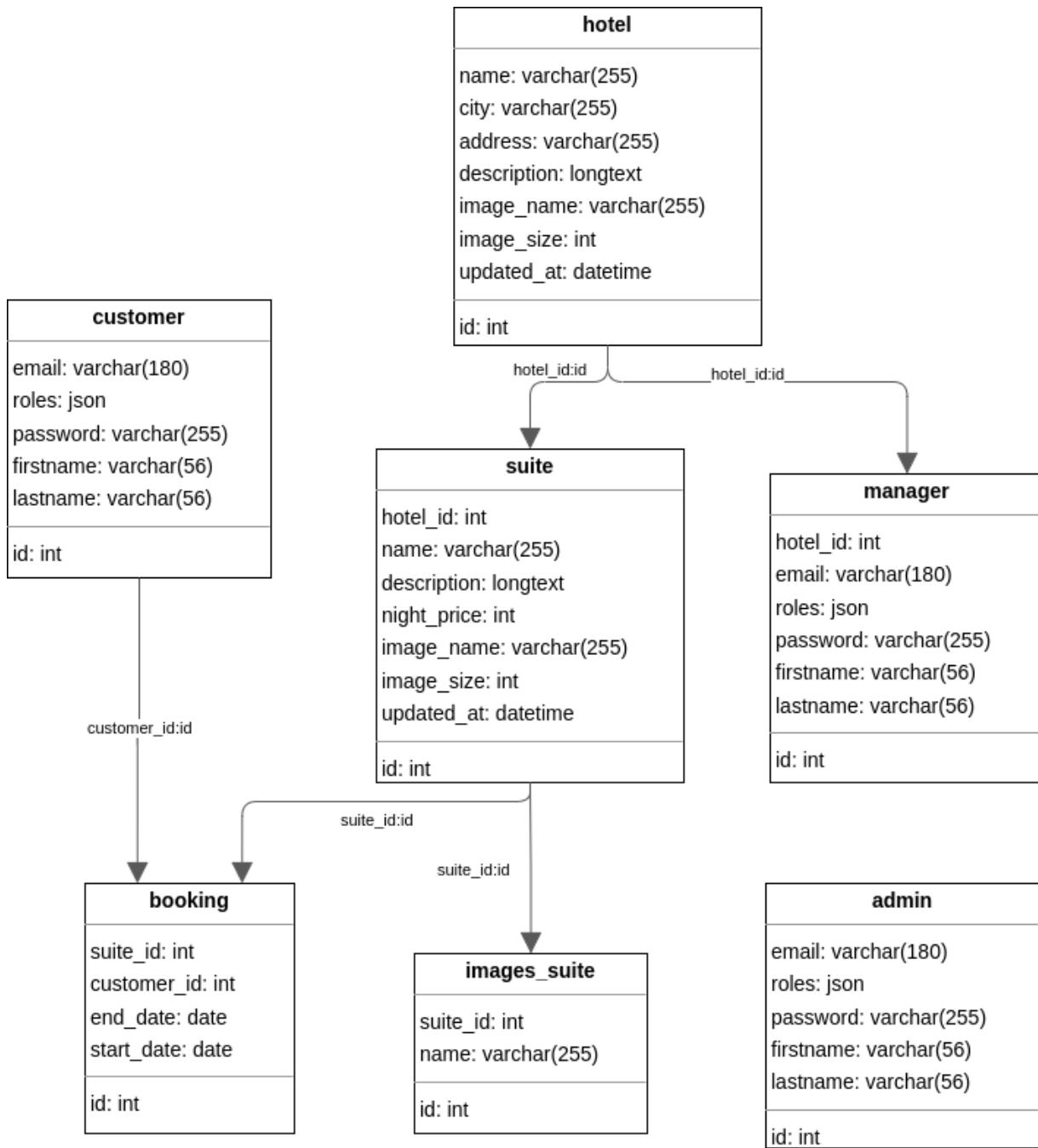


Diagramme de classe



MPD final

Initialisation du projet :

Symfony est un framework PHP ayant l'avantage d'être fourni avec énormément d'outils de sécurité, et de bundles permettant de créer une application simple et sécurisée qui correspond à la demande du client.

C'est l'ORM Doctrine qui fait la relation entre Symfony et la base de données.

Doctrine s'installe automatiquement lors de la création d'une application Symfony avec le flag `-webapp` :

```
symfony new my_project_directory -webapp
```

Création de la base de donnée :

Le fichier `.env` situé à la racine du projet contient les informations importantes comme l'accès à la base de données

Pour des raisons de sécurité, la documentation Symfony conseille de ne pas modifier le fichier `.env` mais d'utiliser un fichier `.env.local` qui ne sera pas commiter sur le serveur.

```
##> doctrine/doctrine-bundle ##>
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
DATABASE_URL="mysql://ug051rwpj8uj81fv:p2i05kldutchdyqo@bv2nbiitj341.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306/fza81c8i9amzwvvh"
# DATABASE_URL="postgresql://symfony:ChangeMe@127.0.0.1:5432/app?serverVersion=13&charset=utf8"
##< doctrine/doctrine-bundle ##<
```

Si la base de données n'est pas créée, il faut lancer la commande :

```
php bin/console doctrine:database:create
```

Création des entités

Dans Symfony, il est nécessaire de créer des entités qui représentent les tables dans la base de données.

```
php bin/console make:entity
```

Avec Symfony, c'est l'ORM Doctrine qui se charge de faire le lien entre notre application et la base de données.

Cela me permet de rester concentré sur l'application et évite l'écriture de nombreuses requêtes SQL.

La console Symfony (CLI) permet de générer très rapidement les scripts SQL.

Pour cela, une fois nos entités créées, il est nécessaire de lancer la commande suivante :

```
php/bin console make:migration
```

Cela génère un fichier de migration comprenant les requêtes SQL à exécuter mais aussi les requêtes inverses (en cas d'erreur, permet d'annuler les requêtes) pour un retour en arrière en sécurité.

C'est toujours bien de contrôler le fichier de migration, cela permet de voir si l'on n'a pas fait d'erreur ou d'oubli :

Par exemple un «NOT NULL» au lieu d'un «NULL».

Dans ce cas, il faut retourner dans le fichier de l'entité concernée, faire la modification et relancer la commande `php/bin console make:migration`

C'est mieux que de modifier le fichier de migration directement, surtout si d'autres personnes travaillent ou reprennent le projet.

```
1 declare(strict_types=1);
2
3 namespace DoctrineMigrations;
4
5 use Doctrine\DBAL\Schema\Schema;
6 use Doctrine\Migrations\AbstractMigration;
7
8 /**
9  * Auto-generated Migration: Please modify to your needs!
10 */
11 final class Version20220331125841 extends AbstractMigration
12 {
13     public function getDescription(): string
14     {
15         return '';
16     }
17
18     public function up(Schema $schema): void
19     {
20         // this up() migration is auto-generated, please modify it to your needs
21         $this->addSql('CREATE TABLE `admin` (`id` INT AUTO_INCREMENT NOT NULL, `email` VARCHAR(180) NOT NULL, `roles` JSON NOT NULL,
22                         `password` VARCHAR(255) NOT NULL, `firstname` VARCHAR(56) NOT NULL, `lastname` VARCHAR(56) NOT NULL, UNIQUE INDEX UNIQ_880E0D76E7927C74 (`email`),
23                         PRIMARY KEY(`id`)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
24         $this->addSql('CREATE TABLE `customer` (`id` INT AUTO_INCREMENT NOT NULL, `email` VARCHAR(180) NOT NULL, `roles` JSON NOT NULL,
25                         `password` VARCHAR(255) NOT NULL, `firstname` VARCHAR(56) NOT NULL, `lastname` VARCHAR(56) NOT NULL, UNIQUE INDEX UNIQ_81398E09E7927C74 (`email`),
26                         PRIMARY KEY(`id`)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
27         $this->addSql('CREATE TABLE `manager` (`id` INT AUTO_INCREMENT NOT NULL, `email` VARCHAR(180) NOT NULL, `roles` JSON NOT NULL,
28                         `password` VARCHAR(255) NOT NULL, `firstname` VARCHAR(56) NOT NULL, `lastname` VARCHAR(56) NOT NULL, UNIQUE INDEX UNIQ_FA2425B9E7927C74 (`email`),
29                         PRIMARY KEY(`id`)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
30     }
31
32     public function down(Schema $schema): void
33     {
34         // this down() migration is auto-generated, please modify it to your needs
35         $this->addSql('DROP TABLE `admin`');
36         $this->addSql('DROP TABLE `customer`');
37         $this->addSql('DROP TABLE `manager`');
38     }
}
```

Exemple d'un fichier migration

Ensuite, il faut exécuter les requêtes SQL afin de les persister en BDD.

La commande est :

```
php bin/console doctrine:migrations:migrate
```

Authentification des utilisateurs :

Le bundle Security permet de sécuriser l'accès à notre application avec 2 principaux concepts:

- l'authentification
- l'autorisation

L'authentification permet de vérifier si l'utilisateur est simple visiteur ou un utilisateur identifié.

L'autorisation, qui intervient après l'authentification, permet d'accorder l'accès au contenu ou non suivant le rôle attribué.

J'ai d'abord créer les utilisateurs en utilisant la commande :

```
php bin/console make:user
```

Un formulaire à remplir demande les informations comme le nom de la classe (ex administrateur), si l'on veut stocker les données en BDD, la propriété qui deviendra l'identification unique pour l'utilisateur (ex email) et si l'on veut hacher les mots de passe en base de données (oui, plus que recommandé !!)

Afin de créer notre premier administrateur en BDD, il est nécessaire d'écrire une requête SQL comme par exemple :

```
INSERT INTO admin (id, email, roles, password, firstname, lastname) VALUES (NULL,  
'admin@test.fr', '[ "ROLE_ADMIN"]',  
'$2y$13$vr1llFEr35ilxDHeuszJGeNdfivcBaKB8XpcwaoyqYiAOrlJMCOny', 'Administrateur',  
'Principal');
```

Puis je créer un formulaire de connexion sécurisé avec la commande :

```
php bin/console make:auth
```

Dans le cas où plusieurs types d'utilisateurs utilisent le même formulaire de connexion, il est nécessaire de configurer le fichier `security.yaml` et lier les différents providers entre eux grâce au `chain_provider`

Sécurité de la base de donnée

Après m'être assuré que le compte admin (généralement root) de la base de données est bien sécurisé (mot de passe présent et respectant les consignes de sécurité) j'ai créé un utilisateur dédié à cette BDD.

Il est impératif de chiffrer (hasher) tous les mots de passe en BDD

Il faut aussi sauvegarder régulièrement la base de données, soit manuellement, soit via un script automatisé (CRON)

```
mysqldump --user manager --password=z59Tl79zE --databases hypnos > hypnos.sql
```

Dans le cas d'un serveur de BDD stocké en physique dans l'entreprise, il est important que le serveur soit sécurisé et accessible uniquement aux personnes autorisées.

Back-office

Afin de faciliter la gestion des établissements et gérants par l'administrateur, j'ai créé un back-office.

Pour cela j'ai utilisé EasyAdmin, un bundle fourni par Symfony.

Il dispose de la puissance et de la sécurité des outils de Symfony.

J'ai lancé la commande :

```
symfony console make:admin:dashboard
```

Cela m'a créé un tableau de bord que j'ai dédié à l'administrateur.

Il pourra créer et gérer les managers ainsi que les hôtels.

J'ai choisi la route /admin pour le tableau de bord, route que j'ai sécurisé dans le fichier security.yaml afin que seuls les administrateurs puissent accéder à cette URL.

```
# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/admin, roles: ['ROLE_ADMIN'] }
    - { path: ^/manager, roles: ['ROLE_MANAGER', 'ROLE_ADMIN']}

    # - { path: ^/profile, roles: ROLE_USER }
```

Extrait du fichier security.yaml

```
<?php
namespace App\Controller\Admin;

use ...

class DashboardController extends AbstractDashboardController
{
    #[Route('/admin', name: 'admin')]
    public function index(): Response
    {
        $adminUrlGenerator = $this->container->get(AdminUrlGenerator::class);
        return $this->redirect($adminUrlGenerator->setController(HotelCrudController::class)->generateUrl());
    }

    public
    function configureDashboard(): Dashboard
    {
        return Dashboard::new()
            ->setTitle('Hypnos');
    }

    public
    function configureMenuItems(): iterable
    {
        yield MenuItem::linkToDashboard(label: 'Dashboard', icon: 'fa fa-home');
        yield MenuItem::section();
        yield MenuItem::linkToCrud(label: 'Établissements', icon: 'fas fa-list', entityFqn: Hotel::class);

        yield MenuItem::linkToCrud(label: 'Managers', icon: 'fas fa-list', entityFqn: Manager::class);
        yield MenuItem::section();
        yield MenuItem::linkToRoute(label: 'Inscription Manager', icon: 'fas fa-user-shield', routeName: 'app_manager_registration');
    }
}
```

Fichier PHP du tableau de bord admin

Hypnos

Rechercher

admin@ecf.fr

[Dashboard](#)

[Établissements](#)

[Managers](#)

[Inscription Manager](#)

Hotel

[Créer Hotel](#)

Nom de l'établissement	Description de l'établissement	Adresse	Ville	Image	Manager
L'Imprévu	Voir le contenu	10 rue de la Joie	Marseille		Aucun(e)
L'Intrépide	Voir le contenu	256 rue de l'Aventure	Ajaccio		Aucun(e)
L'Éclectique	Voir le contenu	58 impasse de la Culture	Nantes		Aucun(e)
Le Romantique	Voir le contenu	Avenue des Champs Elysée	Paris		Aucun(e)
Le Discret	Voir le contenu	5 rue de l'Amour	Strasbourg		Tom Jones
Le Fantasque	Voir le contenu	49 rue du carnaval	Lille		Albert Rodriguez
Le Belvédère	Voir le contenu	Le Bout du Monde	Lorient		Aucun(e)

7 résultats

< Précédent 1 Suivant >

Aperçu du tableau de bord administrateur

Pour la gestion des suites par un manager, j'ai créé un CRUD.

```
php bin/console make:crud ManagerSuiteController
```

Accueil Nos établissements Mon compte Contact Déconnexion

Gestions des suites

Nom de la suite	Description	Prix par nuit	Actions
Suite Royale	Est maiores natus sit quos consectetur qui sint labore et eveniet explicabo. A doloribus aliquam et galisum consequuntur eos voluptas soluta qui voluptas necessitatibus aut rerum magni quia aliquam est quia porro.	250 €	Voir Éditer Suppression
Suite Princière	Sed vero corrupti aut cumque doloribus et recusandae labore et enim labore non animi ipsum quo beatiae voluptates sed ullam aliquid. Qui quae dolorem ut mollitia exercitationem aut officia repudianda sit eveniet aspernatur aut corporis ullam et omnis illum. 33 molestiae error est tempora obcaecati ea molestias sequi sit quia tempore.	130 €	Voir Éditer Suppression
Suite Nuptiale	Et aspernatur quia eum consectetur inventore aut temporibus quos qui tempora ipsum? Et incident necessitatibus vel iure galisum et recusandae dolores est internos alias ea voluptatem quia non accusantium ullam qui iste rerum.	120 €	Voir Éditer Suppression

[Ajouter une suite](#)

Version Front-end du CRUD

Front-end :

Concernant le front-end, j'ai utilisé Bootstrap qui est un framework CSS qui permet de disposer d'outils et de composants prêt à l'emploi. Il gère aussi facilement le responsive grâce à un système de grille.

J'ai aussi utilisé des médias queries pour peaufiner mon affichage responsive.

Sur ce projet, c'est la version scss de Bootstrap qui a été utilisée, afin de pouvoir personnaliser et intégrer la charte graphique.

J'ai utilisé l'approche "Mobile first" qui consiste à penser d'abord l'application sur mobile et tablette, de façon qu'elle soit plus légère pour ensuite l'adapter sur des ordinateurs qui pourront afficher plus d'éléments ou d'animations complexes car disposant de plus de puissance.

Twig est un moteur de template (ou gabarit) qui facilite l'intégration front-end ainsi que la communication avec le back-end.

Twig joue aussi un rôle important dans la sécurité de l'application, par défaut il échappe les caractères afin d'éviter l'exécution d'HTML et de JS dans les formulaires par ex.

Il permet via le composant de sécurité de Symfony de contrôler ce que l'on affiche suivant le rôle et l'utilisateur connecté.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary sticky-top">
    <a class="navbar-brand mx-5" href="{{ path('app_home') }}>
        </a>
    <button class="navbar-toggler me-4" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse text-center mx-3" id="navbarSupportedContent">
        <ul class="nav navbar-nav mx-auto mx-auto">
            <li class="nav-item">
                <a href="{{ path('app_home') }}" class="btn btn-outline-light my-1 mx-2">Accueil</a>
            </li>
            <li class="nav-item">
                <a href="{{ path('app_hotel_list') }}" class="btn btn-outline-light my-1 mx-2">Nos établissements</a>
            </li>
        {% if is_granted('ROLE_CUSTOMER') %}
            <li class="nav-item">
                <a href="{{ path('app_customer_view_booking') }}" class="btn btn-outline-light my-1 mx-2">Mon compte</a>
            </li>
            <li class="nav-item">
                <a href="{{ path('app_booking_registration') }}" class="btn btn-outline-light bg-lg my-1 mx-2">Réservation</a>
            <li class="nav-item">
                <a href="{{ path('app_contact_form') }}" class="btn btn-outline-light my-1 mx-2">Contact</a>
            <li class="nav-item">
                <a href="{{ path('app_logout') }}" class="btn btn-outline-light bg-lg my-1 mx-2">Déconnexion</a>
            </li>
        {% elseif is_granted('ROLE_MANAGER') %}
            <li class="nav-item">
                <a href="{{ path('app_hotel_edit') }}" class="btn btn-outline-light my-1 mx-2">Mon compte</a>
            <li class="nav-item">
                <a href="{{ path('app_contact_form') }}" class="btn btn-outline-light my-1 mx-2">Contact</a>
            <li class="nav-item">
                <a href="{{ path('app_logout') }}" class="btn btn-outline-light my-1 mx-2">Déconnexion</a>
            </li>
        {% elseif is_granted('ROLE_ADMIN') %}
            <li class="nav-item">
                <a href="{{ path('admin') }}" class="btn btn-outline-light my-1 mx-2">Mon compte</a>
            <li class="nav-item">
                <a href="{{ path('app_contact_form') }}" class="btn btn-outline-light my-1 mx-2">Contact</a>
            <li class="nav-item">
                <a href="{{ path('app_logout') }}" class="btn btn-outline-light my-1 mx-2">Déconnexion</a>
            </li>
        {% else %}
            <li class="nav-item">
                <a href="{{ path('app_login') }}" class="btn btn-outline-light my-1 mx-2">Connexion</a>
            <li class="nav-item">
                <a href="{{ path('app_customer_registration') }}" class="btn btn-outline-light my-1 mx-2">Inscription</a>
            <li class="nav-item">
                <a href="{{ path('app_contact_form') }}" class="btn btn-outline-light my-1 mx-2">Contact</a>
            </li>
        {% endif %}</div>
```

Code source de la barre de navigation codée avec Bootstrap.

L'affichage de certaines parties est conditionné par le contrôle du ROLE de l'utilisateur.
Cela permet de personnaliser l'affichage du menu suivant l'utilisateur.

Script JS

Afin de rendre plus dynamique la page de réservation, j'ai ajouté un script JS afin d'afficher le sélecteur des suites ainsi que les suites correspondantes à l'hôtel choisi.

```
</select>
</div>

<div id="select-suite" class="form-group my-3">
    <label for="suite" class="form-label">Choisissez la suite désirée</label>
    <select id="suite" name="suite" class="form-control w-50">
        </select>
</div>

<div class="dates mt-4">
    <div class="start_date input-group form-group d-flex align-items-center mb-4 w-50">
        <label for="startdate_datepicker" class="form-label">Date de début du séjour :</label>
        <input class="form-control start_date ms-3" type="date" name="startDate" placeholder="Date de début du séjour" value="2023-09-01">
    </div>
    <div class="end_date input-group d-flex align-items-center mb-4 w-50">
        <label for="enddate_datepicker" class="form-label">Date de fin du séjour</label>
        <input class="form-control end_date ms-3" type="date" name="endDate" placeholder="Date de fin du séjour" value="2023-09-05">
    </div>
</div>

<button type="submit" id="submit" class="btn btn-primary my-4">Valider la réservation</button>
{{form_end(form)}}
{%- if notification %}
    <div class="alert alert-success w-75" id="alert" role="alert">
        {{ notification }}
    </div>
{%- endif %}
```

Extrait de l'HTML

D'abord on cache le sélecteur de suite au chargement de la page:

```
$('#select-suite').hide();
```

Ensuite, au changement de valeur du select, plusieurs opérations :

On récupère la valeur de l'hôtel :

```
let hotel_id = $('#hotel').val();
```

On enlève la div #alert (message de notification) :

```
$('#alert').remove();
```

Vide les champs du select suite :

```
$('#suite').empty();
```

Puis on récupérer les données des suites dans un controller dédié via une fonction asynchrone dans laquelle on passe hotel ID en paramètre

```
let selectedSuites = await getSuites(hotel_id);
```

```
#[Route('/suite/api', name: 'app_api', methods: ["GET"])]
public function findSuite(SuiteRepository $suiteRepository)
{
    $listeSuite = $suiteRepository->findAll();
    $listeResponse = array();
    foreach ($listeSuite as $suite) {
        $listeResponse[] = array(
            'id' => $suite->getId(),
            'name' => $suite->getName(),
            'night_price' => $suite->getNightPrice(),
            'hotel_id' => $suite->getHotel()->getId(),
        );
    }
    $response = new Response();
    $response->setContent(json_encode(array('suite'=>$listeResponse)));
    $response->headers->set("Content-Type", "application/json");
    $response->headers->set("Access-Control-Allow-Origin", "*");
    return $response;
}
```

La réponse sera ensuite filtrée dans la fonction afin de ne retourner que les suites correspondantes à l'hôtel choisi.

```
// Fonction pour récupérer les données des suites dans un controller dédié
async function getSuites(hotel_id) {
    let response = await fetch(`{{ path('app_api') }}`, {method: 'get'});
    if (response.status === 200) {
        let data = await response.json();

        let filterResponse = [];
        for (let i = 0; i < data.suite.length; i++) {
            if (data.suite[i]['hotel_id'] == hotel_id) {
                filterResponse.push(data.suite[i]);
            }
        }
        return filterResponse;
    }
}
```

Ensuite on boucle sur la réponse pour afficher les valeurs des options

```
selectedSuites.forEach(suite => {
    $('#suite').append($('<option>', {
        value: suite.id,
```

```

        text: suite.name
    } )
})

```

Le sélecteur s'affichera au bout de 1,5 seconde le temps que la requête s'exécute.

Si la valeur du sélecteur Hotel = 0, on supprime le sélecteur de suite

<option value="0">---Choisissez votre hotel---</option>

```

setTimeout ("$( '#select-suite' ).show()", 1500 );
if ($( '#hotel' ).val() === '0' ) {
    $('#select-suite').hide();
}

```

```

// Désactive le select suite au chargement
$('#select-suite').hide();
// fonction d'affichage du select suite et de ses données
$('#hotel').on('change', function () {
    let hotel_id = $('#hotel').val();
    $('#alert').remove();
    $('#suite').empty();
    async function selectSuites() {
        let selectedSuites = await getSuites(hotel_id);
        selectedSuites.forEach(suite => {
            $('#suite').append($('', {
                value: suite.id,
                text: suite.name
            }))
        })
    }
    selectSuites();
    setTimeout("$('#select-suite').show()",1500);
    if ($('#hotel').val() === '0' ) {
        $('#select-suite').hide();
    }
})

```

Manuel d'utilisation

Afin d'aider à la prise en main de l'application par les utilisateurs, j'ai créé un manuel d'utilisation présentant les fonctionnalités et leur utilisation. (voir extrait en Annexes page :)

Référencement :

Avant la mise en ligne du site, j'ai optimisé le référencement en utilisant des balises méta dans la partie «Head» de mon code :

Meta description : Description du site (80 à 160 caractères maxi)

Meta keywords : Liste de mots clés afférents au contenu du site

Ces balises permettent aux moteurs de recherche de référencer le site.

A noter que la balise méta Keywords est dépréciée par beaucoup de moteurs de recherche comme Google depuis longtemps, mais est toujours utilisée par d'autres.

7. Jeu d'essai :

Fonctionnalité la plus représentative : La gestion des suites

L'application présente un ensemble d'hôtels et leurs suites. Ce sont les managers qui se chargent de la gestion des suites.

Ce jeu d'essai à pour but de tester l'ensemble des opérations possibles :

- Création, Visualisation, mise à jour , suppression d'une suite (CRUD)
- Ajout et suppression d'une galerie de photos
- Ajout d'une image à la Une

Le manager se connecte avec ses identifiants fourni par l'administrateur.

Si aucun hôtel ne lui est attribué, un message d'alerte apparaît :

The screenshot shows a dark blue header bar with the Hypnos logo on the left and five buttons on the right: Accueil, Nos établissements, Mon compte, Contact, and Déconnexion. Below the header is a light gray content area containing a message: "Aucun hôtel ne vous est encore assigné, merci de patienter quelques minutes." This indicates that the manager has not been assigned any hotel yet and should wait a few minutes.

Si un hôtel lui a été attribué le manager est redirigé sur la page de son établissement, il a alors accès à la page de gestion des suites :

Gestions des suites

Nom de la suite	Description	Prix par nuit	Actions
Suite Royale	Quo nemo dolor et sapiente amet sit vero sunt et maiores quidem et facilis laboriosam. Et impedit assumenda ut Quis magni et exercitationem optio aut nihil voluptate quo ipsam voluptatum eos quaerat soluta qui repellat quasi.	200 €	Voir Éditer Suppression
Suite Princière	Qui accusamus galisum et totam provident At doloremque quia! Vel nihil rerum qui atque culpa rem itaque alias hic adipisci voluptatem. Non iusto doloremque qui cumque nisi et ducimus natus ea voluptates dolor aut aliquid porro ea illum voluptatibus aut tempora pariatur.	100 €	Voir Éditer Suppression

[Ajouter une suite](#)

© 2004-2022 Hypnos

Les destinations idéales pour les amoureux

[CGV](#) | [RGPD](#)

Ajout d'une suite :

J'ai rempli le formulaire dédié avec les informations, 1 image à la une et 2 images dans la galerie:

Créer une nouvelle suite

Nom de la suite

Suite Nuptiale

Description de la suite

Jeu d'essai

Prix de la nuitée (€)

810

[Valider les informations](#)

[Retour à la liste des suites](#)

Image en avant de la suite

[Choisir un fichier](#) greece-844269_1920.jpg

[Valider l'image en avant](#)

Images de la suite

Sélect. fichiers 2 fichiers

[Valider les images de la suite](#)

[Retour à la liste des suites](#)

© 2004-2022 Hypnos

Les destinations idéales pour les amoureux

[CGV](#) | [RGPD](#)

Après soumission du formulaire, la suite apparaît bien dans la liste des suites, sur le site ainsi qu'en base de données.

17	17	7 Suite Royale	Et laboriosam ut quis galisum eum rerum itaque. Et n...	150 625bf37e64108954668702.jpg	260270 2022-04-17 13:01:18
18	18	7 Suite Princière	Cum consequatur ut quia quidem et saepe ipsum. Hic d...	140 625bf351e848c011236616.jpg	262403 2022-04-17 13:00:33
19	21	6 Suite Nuptiale	Jeu d'essai	810 627d3ce904494572927078.jpg	1280646 2022-05-12 16:59:21

Table suite

WHERE		ORDER BY id DESC		
		id	suite_id	name
1	15	21	8081a733c1886abf48f4ffbba4f77f2f.jpg	
2	14	21	6d54985952f61e146542900eec86b18d.jpg	
3	11	15	18c7ab993a61f3808964961c203e5a6f.jpg	
4	10	12	09edab0d415f59d207ba04389e59dc8e.jpg	

Table images_suite

De la page gestions de suites, j'ai plusieurs action possible:

- Voir : permet d'afficher l'ensemble des informations et photos de la suite
- Editer : Un formulaire identique à celui de création permet de modifier des informations, ajouter ou supprimer des photos
- Suppression de la suite : permet comme son nom l'indique de supprimer la suite.

Après la suppression de la suite, je vérifie non seulement que la suite est bien supprimée de la BDD mais aussi les images correspondantes dans la table images_suites ainsi que la disparition physiques des photos sur le serveur.

7. Veille de sécurité

Même si Symfony est fourni avec énormément d'outils et est sécurisé de base, comme toute application contenant du code, il reste vulnérable aux attaques.

Concernant Symfony, SensioLabs son éditeur tiens un blog concernant les vulnérabilités rencontrées depuis la version 1.0 (<https://symfony.com/blog/category/security-advisories>)

Dans la documentation Symfony, un gros chapitre est dédié à la sécurité et au bundle Security :

<https://symfony.com/doc/current/security.html>

Plusieurs moyens de connexion et d'accès sont présentés, et des outils de protections comme par exemple la limitation du nombre d'essais de connexion (Limiting Login Attempts)

Il est important d'aller régulièrement sur le site du CERT-FR (<https://www.cert.ssi.gouv.fr/>) et de l'Agence nationale de la sécurité des systèmes d'information (<https://www.ssi.gouv.fr/>) afin de connaître les menaces et vulnérabilité détectées récemment.

L'ANSII met aussi à disposition des guides de bonnes pratiques concernant la sécurité des sites web, des réseaux...

Le plus intéressant pour moi en tant que développeur est le guide “Sécuriser un site web” :

<https://www.ssi.gouv.fr/administration/guide/recommandations-pour-la-securisation-des-sites-web/>

Ce guide explique les principales menaces et attaques envers les site web comme les attaques Cross-Site Scripting et Cross-Site Request Forgery (CRSF)

Plus globalement, la fondation Owsap, qui travaille à améliorer la sécurité des applications web, met aussi à disposition des outils et formations liés à la sécurité.

Ils publient chaque années un relevé des failles majeures rencontrées dans les applications ainsi que leur évolution au fil du temps.

<https://owasp.org/www-project-top-ten>

Et surtout la fondation met à disposition des aides-mémoires complets sur les principales menaces et attaques ainsi que les moyens de s'en prémunir.

<https://cheatsheetseries.owasp.org/>

8. Recherche et traduction anglophone

Afin d'identifier rapidement un hôtel ou une suite, je devais mettre en avant une image.

Pour cela j'ai rechercher sur la documentation de Symfony comment uploader une image et la gérer via un FormType.

Symfony via sa documentation (How to Upload Files) informe que suivant les besoins, il est possible d'utiliser un bundle qui propose des opérations de base (comme la création, le renommage et la suppression) et qui s'intègre parfaitement à Doctrine, l'ORM utilisé par Symfony. Il s'agit du bundle VichUploaderBundle.

Si la mise en place du bundle n'est pas compliquée, certaines étapes demandent de l'attention.

The final step is to create a link between the filesystem and the entity you want to make uploadable.

We already created an abstract representation of the filesystem (the mapping), so we just have to tell the bundle which entity should use which mapping. In this guide we'll use annotations to achieve this, but you can also use YAML or XML.

First, annotate your class with the Uploadable annotation. This is really like a flag indicating that the entity contains uploadable fields.

Next, you have to create the two fields needed for the bundle to work:

create a field (e.g. imageName) which will be stored to the database as a string. This will hold the filename of the uploaded file.

create another field (e.g. imageFile). This will store the UploadedFile object after the form is submitted. This should not be persisted to the database, but you do need to annotate it.

Traduction :

Cette étape finale permet de créer un lien entre le système de fichier et l'entité que vous voulez rendre téléchargeable (transférable),

Nous avons déjà créé une représentation abstraite du système de fichiers (le mappage), donc il nous reste juste à indiquer au bundle avec quelle entité nous devons lier ce mappage.

Dans ce guide, nous utiliserons les annotations pour y parvenir mais vous pouvez aussi utiliser YAML ou XML.

D'abord, nous allons annoter notre classe avec l'annotation <Uploadable>. C'est comme un drapeau qui indique que l'entité contient des champs téléchargeables.

Ensuite, vous avez à créer 2 champs nécessaire au bon fonctionnement du bundle :

- Créer un champ (ex imageName) qui sera persisté dans la BDD comme « string »

Il contiendra le nom du fichier téléchargé.

-Créer un autre champ (ex `imageFile`), celui ci stockera l'objet « `UploadedFile` » après que le formulaire soit soumis.

Celui-ci ne sera pas persisté dans la BDD mais vous devez quand même l'annoter.

Nb : Mapping n'a pas de véritable traduction officielle en français car a de multiple usage en anglais mais il est d'usage d'utiliser le mot mappage.

Mappage : Association des données appartenant à plusieurs ensembles différents (modèle logique de données, base de données de production...)

Conclusion

Concernant l'application, elle est fonctionnelle mais pourra être mise à jour avec d'autres fonctionnalités comme le contrôle des dates de réservations, le pré-remplissage automatique des champs du formulaire de réservation et aussi l'affichage des réservations dans le back-Office Manager.

Ce projet fictif m'a permis de mettre en pratique les connaissances acquises lors de la formation et de mieux appréhender les différentes facettes du métier de développeur.

J'ai aussi réalisé pendant ce projet une veille permanente afin de compléter mes connaissances avec l'aide des documentations officielles, de forums dédiés, de tutoriels vidéos,...

Cela m'a permis de me confronter aux problèmes que peuvent rencontrer les développeurs (bugs, problèmes de déploiement) et m'a montré la nécessité réelle de pratiquer.

Annexes

Hypnos

Accueil | Nos établissements | Connexion | Inscription | Contact

Liste de nos établissements

Hotel L'Imprévu ♥ Marseille



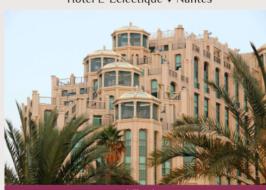
Voir l'hôtel

Hotel L'Intrépide ♥ Ajaccio



Voir l'hôtel

Hotel L'Éclectique ♥ Nantes



Voir l'hôtel

Hotel Le Romantique ♥ Paris



Voir l'hôtel

Hotel Le Discret ♥ Strasbourg



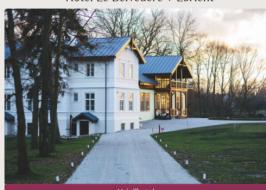
Voir l'hôtel

Hotel Le Fantasque ♥ Lille



Voir l'hôtel

Hotel Le Belvédère ♥ Lorient



Voir l'hôtel

© 2004-2022 Hypnos   
Les destinations idéales pour les amoureux
CGV | RGPD

Version Desktop de la page Nos établissements

Hypnos

☰

Liste de nos établissements

Hotel L'Imprévu ♥ Marseille



Voir l'hôtel

Hotel L'Intrépide ♥ Ajaccio

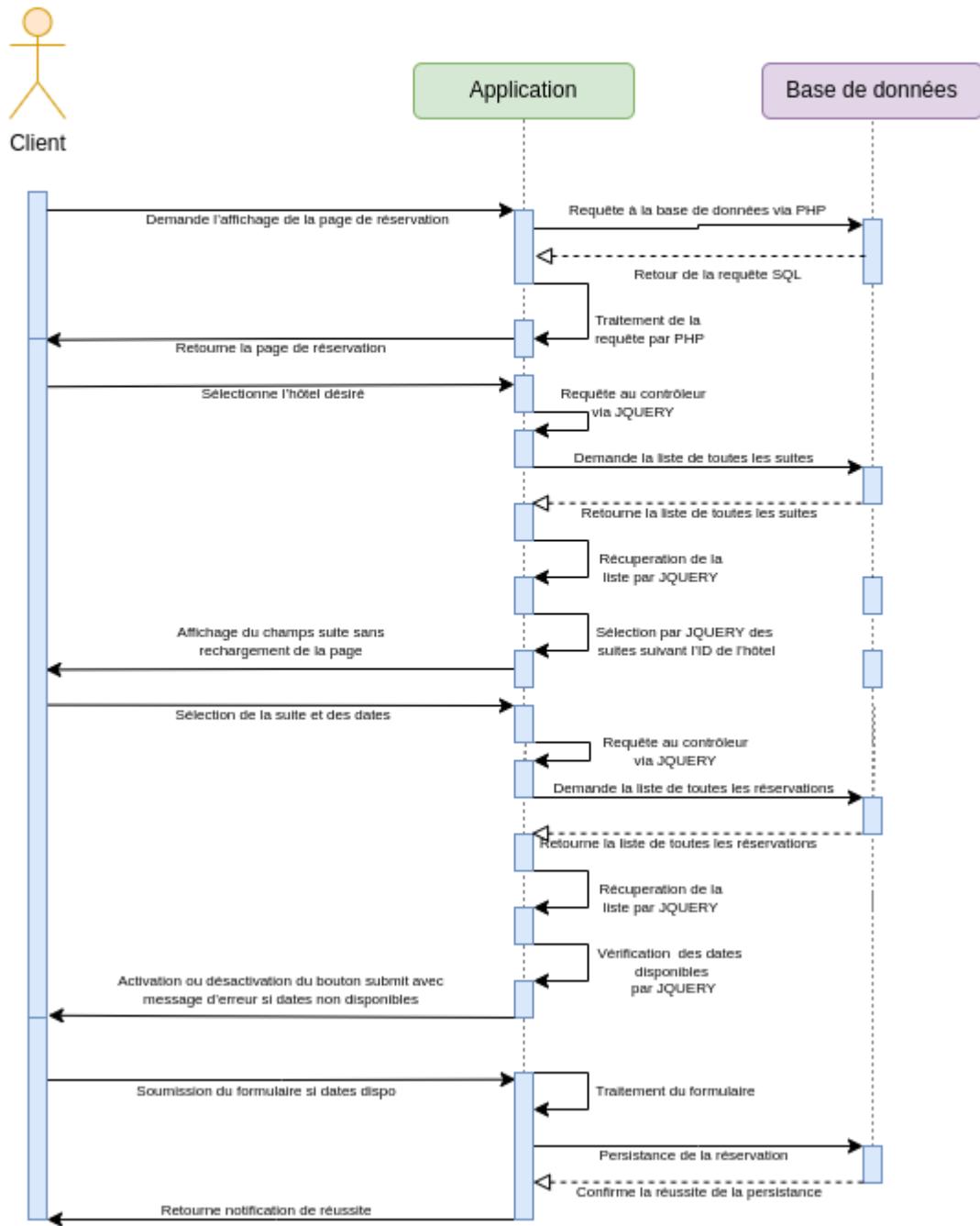


Voir l'hôtel

Version Mobile

Diagramme de séquence

Diagramme de séquence - Réserver une suite



Créer et modifier les établissements

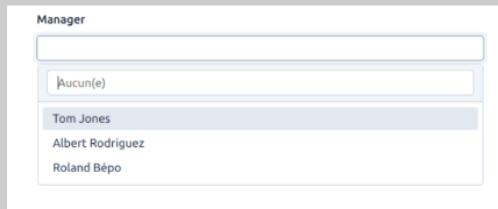
Assigner un gérant

Pour assigner un gérant à un hôtel, il faut que le gérant soit inscrit au préalable.

L'assignation d'un gérant peut se faire soit à la création, soit pendant la modification d'un hôtel.

Pour cela cliquez sur le champ Manager, la liste des gérants va apparaître.

Vous pouvez soit choisir un gérant soit enlever un précédent gérant en cliquant sur «Aucun(e)»



Champ choix du gérant

Sauvegarder et continuer l'édition | Sauvegarder les modifications

Nom de l'établissement*
Le Discret

Description de l'établissement*
Et laborum quis est beatae voluptas ut cumque inventore id reciendis cupiditate ut provident voluptateb. Qui reprehenderit nemo qui accusamus eamur et dolore corrupti non voluptatibus deleniti aut aliquam exercitationem! Est neque internos eum provident iure et sequi perferendis ea similique eligendi ut dolore ipsum hic rerum.

Adresse*
5 rue de l'Amour

Ville*
Strasbourg

Image

Choisir un fichier | Supprimer?

Manager
Tom Jones

Ecran modification d'un hôtel

14

Extrait du manuel d'utilisation